Lab6

Pass

Ahmed Alhasan, Yash Pawar, Mohsen Pirmoradiyan

3/6/2020

Question 1: Genetic algorithm

1. Define the function

$$f(x) = \frac{x^2}{e^x} - 2\exp(-(9\sin x)/(x^2 + x + 1))$$

```
f = function(x){
  f=(x^2/exp(x)) - 2*exp(-(9 * sin(x))/(x^2 + x + 1))
  return(f)
}
```

2. Define the function crossover(): for two scalars x and y it returns their "kid" as (x+y)/2.

```
crossover = function(x,y){
  return((x+y)/2)
}
```

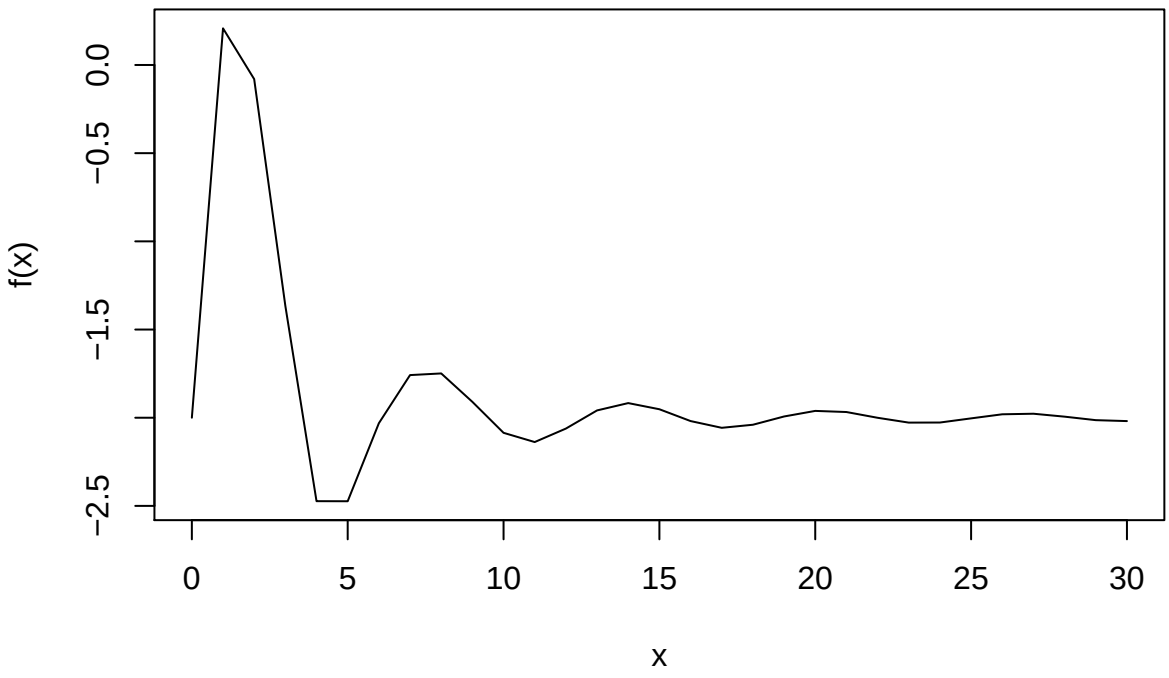
3. Define the function mutate() that for a scalar x returns the result of the integer division x^2 mod 30.

```
mutate = function(x){
  return(x^2 %% 30)

}
```

4. Writing the genetic function with two arguments: *maxiter* **and** *mutprob*

The figure below shows the values of function f for the range (0,30). A maximum can be seen in the plot.

```
x = seq(0,30)
  plot(x, f(x), type = "l")
```

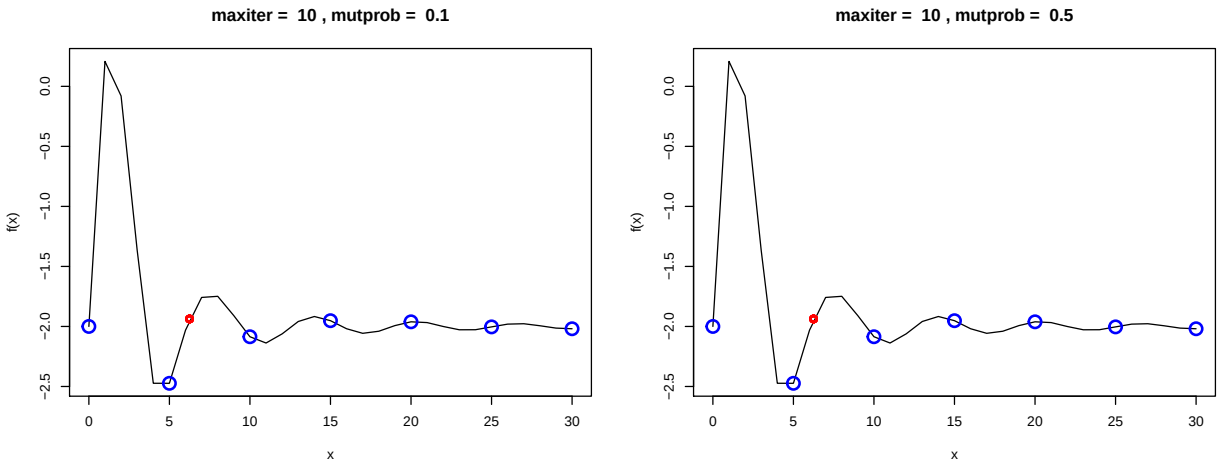


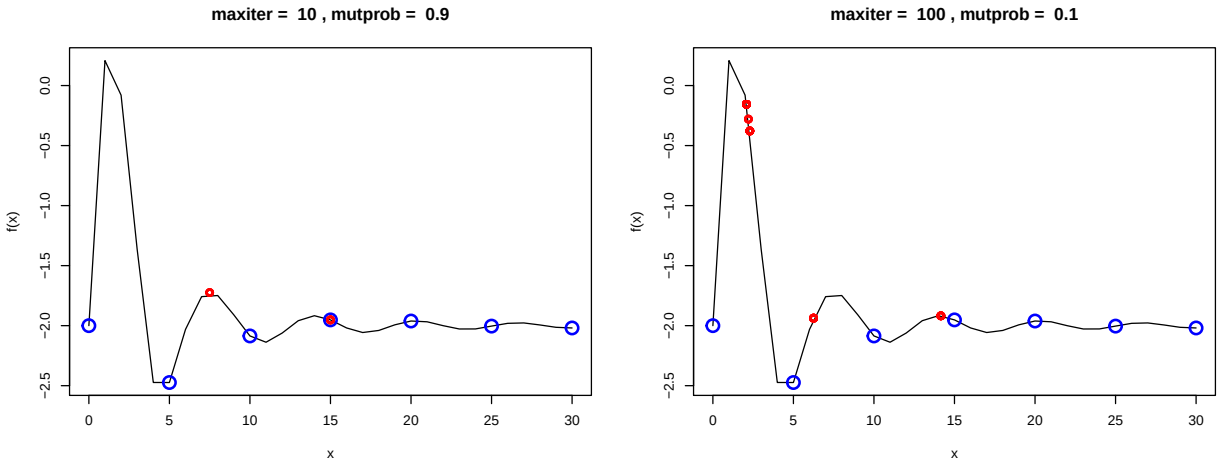

```
genetic.func = function(maxiter, mutprob){
  x = seq(0,30)
  plot(x, f(x), type = "l",
       main = paste("maxiter = ",as.character(maxiter), ", mutprob = ", as.character(mutprob)))

  #Initial population
  X = seq(0,30,5)
  # Function values of the population
  Values = f(X)
  points(X, Values, col="Blue", cex=2, lwd=2)
  Max = c()
  final.X = c()
  for (i in 1:maxiter) {
    #Two indexes are randomly sampled from the current population
    parent.ind = sample(1:length(X), 2)

    # index with the smallest objective function
    victim.ind = which.min(Values)

    # produce new kid by crossovering the parents
    new.kid = crossover(X[parent.ind[1]], X[parent.ind[2]])

    # Mutate this kid with probability mutprob
    new.kid = ifelse(runif(1)>mutprob, mutate(new.kid), new.kid)

    # Replacing victim with the kid
```

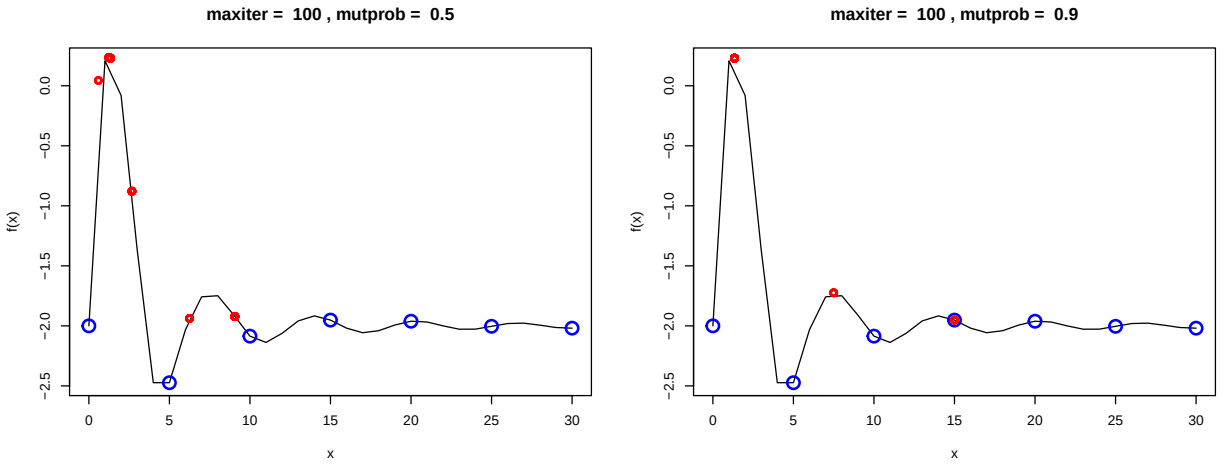
```
    X[victim.ind] = new.kid

    # Updating Values

    Values = f(X)
    final.X = c(final.X, X[which.max(Values)])

    cur.max = max(Values)
    Max = c(Max, cur.max)


    #points(X[which.max(Values)], cur.max, col=i)
    #Sys.sleep(0.1)

  }
  points(final.X, f(final.X), col = "Red", lwd=2)
}
```

5. Run your code with different combinations of maxiter= 10,100 and mutprob= 0.1, 0.5, 0.9

The figures below were generated with different combination of iteration and mutation probability. The seed was set to(12345). The blue circles are the initial population, and the red ones are the generated population by the genetic function. The maximum value can be found when the maxiter is 100. The mutation probability also has an effect on the result.


maxiter = 10 , mutprob = 0.1
maxiter = 10 , mutprob = 0.5
f(x)
x

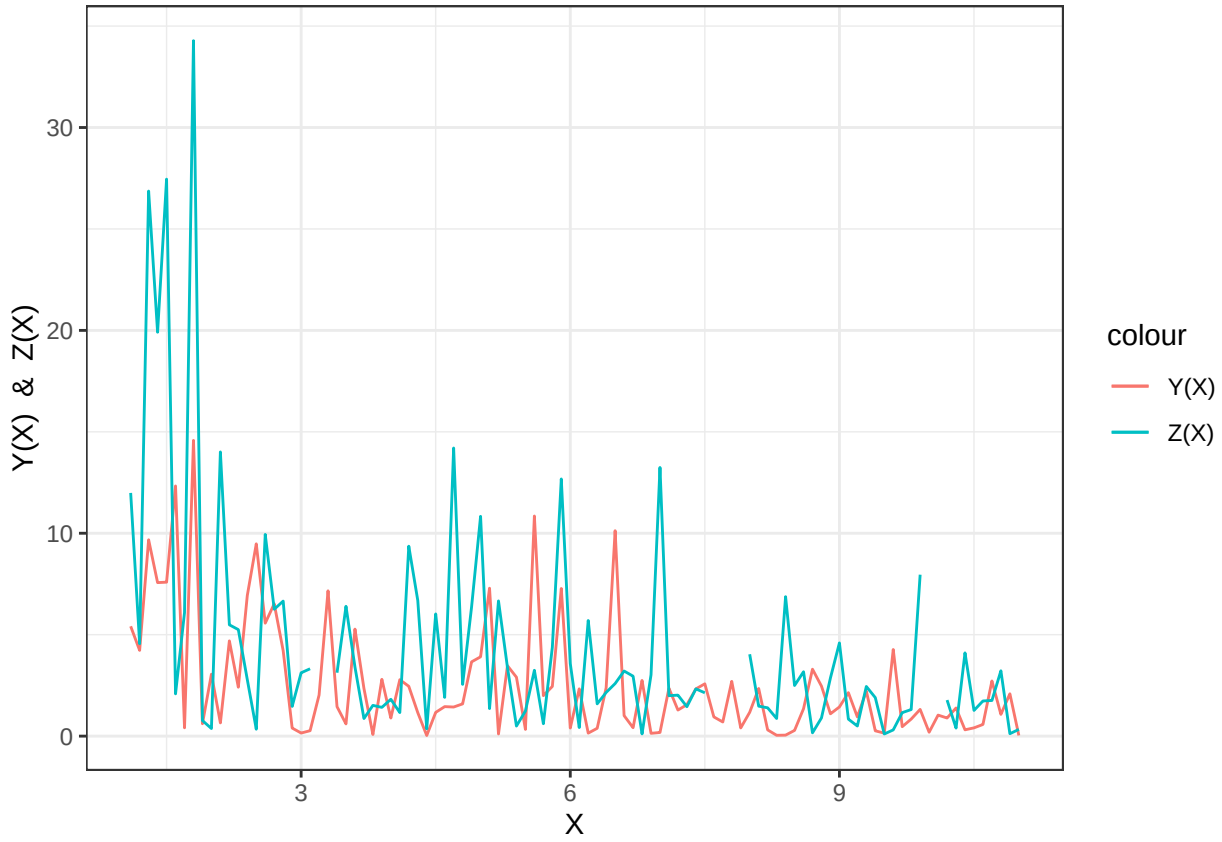

maxiter = 10 , mutprob = 0.9
maxiter = 100 , mutprob = 0.1
f(x)
x


maxiter = 100 , mutprob = 0.5
maxiter = 100 , mutprob = 0.9
f(x)
x

Question 2: EM algorithm

1. Reading the data and Making time series plot

```
Data = read.csv("E:/LiU/2ndSemester/Computational Statistics/Labs/6/physical1.csv")

ggplot(Data)+ geom_line(mapping = aes(X, Y, color="Y(X)"))+ geom_line(mapping = aes(X, Z, color="Z(X)"))
```



From the plot it it seems that the variations witin Y and Z follow each other to some extent. However, they are not highly correlated.

As X increases the variation of both responses decrease.

They are correlated if you "marginalise" over X. But of course you can't "marginalise" X because there is no probability distribution assigned to X here.

2. Dealing with missing values.

The following models must be used:

$$Y_i \sim \exp(X_i/\lambda)$$
$$Z_i \sim \exp(X_i/2\lambda)$$

The goal is to derive an EM algorithm that estimates λ

We know that if X is distributed as exp(λ), then:

$$f(x|\lambda) = \lambda e^{-\lambda x}$$

But X is not distributed as Exp(lambda). X does not have a distribution as the model specified in the question deems it as constant instead of random variable.

Using this distribution we have:

$$f(Y|X,\lambda) = \frac{X}{\lambda}\exp(-XY/\lambda)$$

and

$$f(Z|X,\lambda) = \frac{X}{2\lambda}\exp(-XZ/2\lambda)$$

Having these two density functions, we now can calculate the likelihood of their joint distribution:

$$L(\lambda) = \prod_{i=1}^{n}\frac{X_i}{\lambda}\exp(-X_iY_i/\lambda)\prod_{i=1}^{n}\frac{X_i}{2\lambda}\exp(-X_iZ_i/2\lambda)$$

$$= \prod_{i=1}^{n}\frac{X_i^2}{2\lambda^2}\exp(-X_iY_i/\lambda)\exp(-X_iZ_i/2\lambda)$$

$$= (\frac{1}{2\lambda^2})^n\exp(-\sum_{i=1}^{n}\frac{X_iY_i}{\lambda}+\frac{X_iZ_i}{2\lambda})\prod_{i=1}^{n}X_i^2$$

Correct

Taking logarithm of this function, we have:

$$l(\lambda) = \ln((\frac{1}{2\lambda^2})^n) + \ln(\exp(-\sum_{i=1}^{n}\frac{X_iY_i}{\lambda}+\frac{X_iZ_i}{2\lambda})) + \sum_{i=1}^{n}\ln(X_i^2)$$

$$= -2n\ln(\lambda) - \ln(2) - \sum_{i=1}^{n}\frac{X_iY_i}{\lambda} - \sum_{i=1}^{n}\frac{X_iZ_i}{2\lambda} + \sum_{i=1}^{n}\ln(X_i^2)$$

$$= -2n\ln(\lambda) - \sum_{i=1}^{n}\frac{X_iY_i}{\lambda} - \sum_{i=1}^{n}\frac{X_iZ_i}{2\lambda} - \ln(2) + \sum_{i=1}^{n}\ln(X_i^2)$$

This function is the log-likelihood function.

The variable Z has some missing values. Thus we divide it into two parts:

$$\sum_{i=1}^{n}\frac{X_iZ_i}{2\lambda} = \sum_{Observed}\frac{X_iZ_i}{2\lambda} + \sum_{Missing}\frac{X_jZ_j}{2\lambda}$$

Hence:

$$l(\lambda) = -2n\ln(\lambda) - \sum_{i=1}^{n}\frac{X_iY_i}{\lambda} - \sum_{Observed}\frac{X_iZ_i}{2\lambda} - \sum_{Missing}\frac{X_jZ_j}{2\lambda} - \ln(2) + \sum_{i=1}^{n}\ln(X_i^2)$$

???

The expectation can now be calculated as:

$$E(l(\lambda)) = -2n\ln(\lambda) - \sum_{i=1}^{n}\frac{X_iY_i}{\lambda} - \sum_{i\in Observed}\frac{X_iZ_i}{2\lambda} - \sum_{j\in Missing}\frac{X_jZ_j}{2\lambda} - \ln(2) + \sum_{i=1}^{n}\ln(X_i^2)$$

Z_j is a missing value so we can assign it the expected value. Since it has an exponential distribution the expected value of Z is:

$$E(Z_j) = \frac{2\lambda_t}{X_j}$$

Where λ_t is the last value that we have for λ.

Thus:

$$E(\sum_{j\in Missing}\frac{X_jZ_j}{2\lambda}) = \sum_{j\in Missing}\frac{X_j2\lambda_t}{2X_j\lambda} = M\lambda_t/\lambda$$

Good

Where M is the number of missing values.

Therefor:

$$E(l(\lambda)) = -2n\ln(\lambda) - \sum_{i=1}^{n}\frac{X_iY_i}{\lambda} - \sum_{i\in Observed}\frac{X_iZ_i}{2\lambda} - M\frac{\lambda_t}{\lambda} - \ln(2) + \sum_{i=1}^{n}\ln(X_i^2), \qquad (1)$$

For the M-step we have to take the derivative of this function and equal it to zero to find the estimation of λ.

$$\frac{\partial(E(l(\lambda)))}{\partial(\lambda)} = \frac{-2n}{\lambda} + \frac{1}{\lambda^2}\sum_{i=1}^{n}X_iY_i + \frac{1}{2\lambda^2}\sum_{i\in Observed}X_iZ_i + \frac{M\lambda_t}{\lambda^2} = 0$$

$$\Rightarrow \lambda_{t+1} = \frac{\sum_{i=1}^{n}X_iY_i}{2n} + \frac{\sum_{i\in Observed}X_iZ_i}{4n} + \frac{M\lambda_t}{2n}, \qquad (2)$$

Correct

The equation (1) is the E-step, and the equation (2) is the M-step.

3. Implement this algorithm in R, use $\lambda_0 = 100$

We iterate over equation (2 two stepe as long as $|\lambda_{t+1} - \lambda_t| < .001$

```
n = nrow(Data)
ind.mis = which(is.na(Data$Z))
M = length(ind.mis)
ind.obs = which(!is.na(Data$Z))
dif = Inf
lambda=100
it=0
repeat{
  lambda.new = (sum(Data$X*Data$Y) + M*lambda+0.5*sum(Data$X[ind.obs]*Data$Z[ind.obs]))/(2*n)

  dif = abs(lambda.new - lambda)
```

```
  lambda = lambda.new
  it = it+1
  print(lambda)
  if(dif < 0.001)break
}
```

```
## [1] 14.26782
## [1] 10.83853
## [1] 10.70136
## [1] 10.69587
## [1] 10.69566
```

```
cat("\nOptimal Lambda:", lambda)
```

```
##
## Optimal Lambda: 10.69566
```

```
cat("\nNumber of iterations:", it)
```

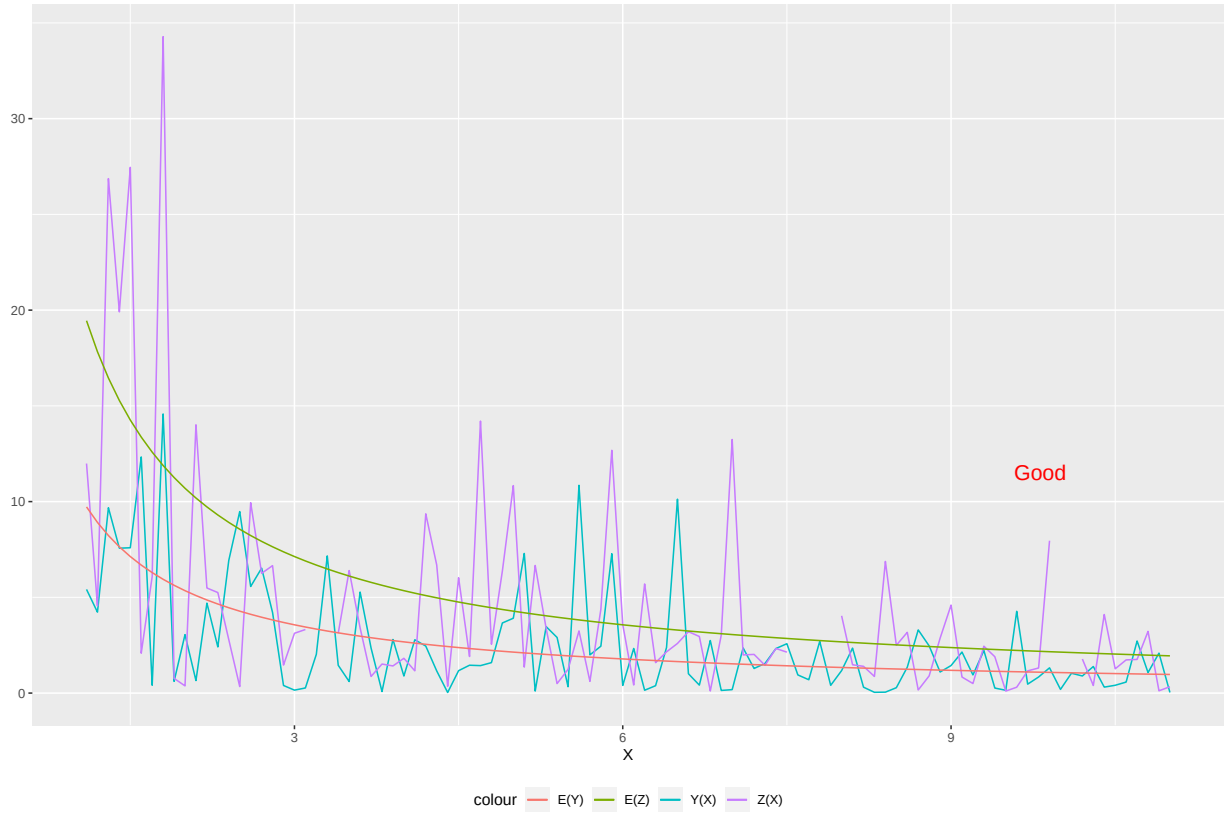
```
##
## Number of iterations: 5
```

4. Plot E [Y] and E [Z] versus X in the same plot as Y and Z versus X. Comment whether the computed λ seems to be reasonable.

$$E(Y) = \frac{\lambda}{X}$$

$$E(Z) = \frac{2\lambda}{X}$$

```
ggplot(Data)+ geom_line(mapping = aes(X, Y, color="Y(X)"))+
  geom_line(mapping = aes(X, Z, color="Z(X)"))+
  geom_line(mapping = aes(X, lambda/X, color="E(Y)"))+
  geom_line(mapping = aes(X, 2*lambda/X, color="E(Z)"))+
  ylab("") +theme(legend.position = "bottom")
```



As the plot shows the expected values for Z and Y seem to be reasonable. Therefore the value for lambda is reasonable.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
library(ggplot2)
f = function(x){
  f=(x^2/exp(x)) - 2*exp(-(9 * sin(x))/(x^2 + x + 1))
  return(f)
}
crossover = function(x,y){
  return((x+y)/2)
}
mutate = function(x){
  return(x^2 %% 30)

}
x = seq(0,30)
  plot(x, f(x), type = "l")
genetic.func = function(maxiter, mutprob){
  x = seq(0,30)
  plot(x, f(x), type = "l",
       main = paste("maxiter = ",as.character(maxiter), ", mutprob = ", as.character(mutprob)))

  #Initial population
  X = seq(0,30,5)
  # Function values of the population
  Values = f(X)
  points(X, Values, col="Blue", cex=2, lwd=2)
  Max = c()
  final.X = c()
  for (i in 1:maxiter) {
    #Two indexes are randomly sampled from the current population
    parent.ind = sample(1:length(X), 2)

    # index with the smallest objective function
    victim.ind = which.min(Values)

    # produce new kid by crossovering the parents
    new.kid = crossover(X[parent.ind[1]], X[parent.ind[2]])

    # Mutate this kid with probability mutprob
    new.kid = ifelse(runif(1)>mutprob, mutate(new.kid), new.kid)

    # Replacing victim with the kid
    X[victim.ind] = new.kid

    # Updating Values

    Values = f(X)
    final.X = c(final.X, X[which.max(Values)])

    cur.max = max(Values)
    Max = c(Max, cur.max)
```

```
    #points(X[which.max(Values)], cur.max, col=i)
    #Sys.sleep(0.1)

  }
  points(final.X, f(final.X), col = "Red", lwd=2)
}
maxiter = c(10, 100)
mutprob = c(0.1, 0.5, 0.9)

par(mfrow = c(3,2))
for (it in maxiter) {
  for (prob in mutprob) {
    set.seed(12345)
    genetic.func(it, prob)

  }

}
Data = read.csv("E:/LiU/2ndSemester/Computational Statistics/Labs/6/physical1.csv")

ggplot(Data)+ geom_line(mapping = aes(X, Y, color="Y(X)"))+ geom_line(mapping = aes(X, Z, color="Z(X)"))
n = nrow(Data)
ind.mis = which(is.na(Data$Z))
M = length(ind.mis)
ind.obs = which(!is.na(Data$Z))
dif = Inf
lambda=100
it=0
repeat{
  lambda.new = (sum(Data$X*Data$Y) + M*lambda+0.5*sum(Data$X[ind.obs]*Data$Z[ind.obs]))/(2*n)

  dif = abs(lambda.new - lambda)
  lambda = lambda.new
  it = it+1
  print(lambda)
  if(dif < 0.001)break
}

cat("\nOptimal Lambda:", lambda)
cat("\nNumber of iterations:", it)


ggplot(Data)+ geom_line(mapping = aes(X, Y, color="Y(X)"))+
  geom_line(mapping = aes(X, Z, color="Z(X)"))+
  geom_line(mapping = aes(X, lambda/X, color="E(Y)"))+
  geom_line(mapping = aes(X, 2*lambda/X, color="E(Z)"))+
  ylab("") +theme(legend.position = "bottom")
```